Administración de torneos

Equipos

El API de equipos consiste en la interacción del recurso “teams” con las siguientes especificaciones

API

URI: /teams

Método: POST

Cuerpo:

```
{"name":"TEAM NAME"}
```

Respuesta:

- HTTP Response: 201
    - Header:
        - Location : TEAM-ID
            - TEAM-ID es el ID generado de la base de datos
- HTTP Response: 409
    - En caso de que el equipo ya exista en la base de datos

URI: /teams

Método: GET

Respuesta:

- HTTP: 200
- Cuerpo
    - Arreglo de equipos en la base de datos, en caso de no tener equipos entonces la respuesta es un arreglo vacío

```
[{"id":"33223e-sdfs","name":"TEAM NAME"}, {"id":"33223e-sdfa","name":"TEAM 2"}]
```

URI: /teams/<TEAM-ID>

Método: PATCH

Cuerpo:

```
{"name":"TEAM NAME UPDATE"}
```

Respuesta:

- HTTP: 204
- HTTP: 404
    - En caso de TEAM-ID no exista en la base de datos

URI: /teams/<TEAM-ID>

Método: GET

Cuerpo:

Respuesta:

- HTTP: 200
- HTTP: 404
    - En caso de TEAM-ID no exista en la base de datos

```
{"id":"33223e-sdfs","name":"TEAM NAME"}
```

Pruebas de código

Se crearán pruebas unitarias en las partes principales del código que exista algún tipo de lógica, es decir en las clases Controller y Delegate.

Las pruebas unitarias consisten en corroborar que el código se comporte de acuerdo a las reglas de negocio.

Pruebas mínimas esperada en TeamController

- Al método que procesa la creación de equipo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a TeamDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 201
- Al método que procesa la creación de equipo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a TeamDelegate es el esperado. Simular error de inserción en la base de datos y validar la respuesta HTTP 409

- Al método que procesa la búsqueda de un equipo por ID, validar que el valor que se le transfiera a TeamDelegate es el esperado. Simular el resultado con un objeto y validar la respuesta HTTP 200
- Al método que procesa la búsqueda de un equipo por ID, validar que el valor que se le transfiera a TeamDelegate es el esperado. Simular el resultado nulo y validar la respuesta HTTP 404
- Al método que procesa la búsqueda de equipos. Simular el resultado con una lista de objetos y la respuesta HTTP 200
- Al método que procesa la búsqueda de equipos. Simular el resultado con una lista vacía y la respuesta HTTP 200
- Al método que procesa la actualización de un equipo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a TeamDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 204
- Al método que procesa la actualización de un equipo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a TeamDelegate es el esperado. Simular ID no encontrado en el Delegate y validar que la respuesta de esta función sea HTTP 404

Pruebas mínimas esperada en TeamDelegate

- Al método que procesa la creación de equipo, validar que el valor que se le transfiera a TeamRepository es el esperado. Simular una inserción válida y que la respuesta de esta función sea el ID generado
- Al método que procesa la creación de equipo, validar que el valor que se le transfiera a TeamRepository es el esperado. Simular una inserción fallida y que la respuesta de esta función sea un error o mensaje usando https://en.cppreference.com/w/cpp/utility/expected.html
- Al método que procesa la búsqueda de un equipo por ID, validar que el valor que se le transfiera a TeamRepository es el esperado. Simular el resultado con un objeto y validar el objeto
- Al método que procesa la búsqueda de un equipo por ID, validar que el valor que se le transfiera a TeamRepository es el esperado. Simular el resultado nulo y validar nullptr
- Al método que procesa la búsqueda de equipos. Simular el resultado con una lista de objetos de TeamRepository

- Al método que procesa la búsqueda de equipos. Simular el resultado con una lista vacía de TeamRepository
- Al método que procesa la actualización de un equipo, validar la búsqueda de TeamRepository por ID, validar el valor transferido a Update. Simular resultado exitoso
- Al método que procesa la actualización de un equipo, validar la búsqueda de TeamRepository por ID. Simular resultado de búsqueda no exitoso y regresar error o mensaje usando https://en.cppreference.com/w/cpp/utility/expected.html

Torneos

El API de equipos consiste en la interacción del recurso “tournaments” con las siguientes especificaciones

API

URI: /tournaments

Método: POST

Cuerpo:

```
{"name":"TEAM NAME"}
```

Respuesta:

- HTTP Response: 201
    - Header:
        - Location : TOURNAMENT-ID
            - TOURNAMENT-ID es el ID generado de la base de datos
- HTTP Response: 409
    - En caso de que el torneo ya exista en la base de datos

URI: /tournaments

Método: GET

Respuesta:

- HTTP: 200
- Cuerpo

- Arreglo de equipos en la base de datos, en caso de no tener equipos entonces la respuesta es un arreglo vacío


```
[{"id":"33223e-sdfs","name":"Torneo Verano 2025"}, {"id":"33223f-sdfs",
"name":"Torneo Invierno 20205"}]
```


URI: /tournaments/<TOURNAMENT-ID>

Método: PATCH

Cuerpo:


```
{"name":"TOURNAMENT NAME UPDATE"}
```


Respuesta:

- HTTP: 204
- HTTP: 404
  - En caso de TOURNAMENT-ID no exista en la base de datos

URI: /tournaments/<TOURNAMENT-ID>

Método: GET

Cuerpo:

Respuesta:

- HTTP: 200
- HTTP: 404
  - En caso de TEAM-ID no exista en la base de datos


```
{
    "id":"33223e-sdfs",
    "name":"Torneo Verano 2025",
    “groups”:[
    {
        “name” : “Group A”
        “teams”: [
            {“id” : “team-id”, “name” : “team name”}
```

```
            ]
        }]
}
```

Pruebas de código

Pruebas mínimas esperada en TournamentController

- Al método que procesa la creación de torneo, validar la transformación de JSON al objeto de dominio Tournament y validar que el valor que se le transfiera a TournamentDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 201
- Al método que procesa la creación de torneo, validar la transformación de JSON al objeto de dominio Tournament y validar que el valor que se le transfiera a TournamentDelegate es el esperado. Simular error de inserción en la base de datos y validar la respuesta HTTP 409
- Al método que procesa la búsqueda de un torneo por ID, validar que el valor que se le transfiera a TournamentDelegate es el esperado. Simular el resultado con un objeto y validar la respuesta HTTP 200
- Al método que procesa la búsqueda de un torneo por ID, validar que el valor que se le transfiera a TournamentDelegate es el esperado. Simular el resultado nulo y validar la respuesta HTTP 404
- Al método que procesa la búsqueda de torneos. Simular el resultado con una lista de objetos y la respuesta HTTP 200
- Al método que procesa la búsqueda de torneos. Simular el resultado con una lista vacía y la respuesta HTTP 200
- Al método que procesa la actualización de un torneo, validar la transformación de JSON al objeto de dominio Tournament y validar que el valor que se le transfiera a TournamentDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 204
- Al método que procesa la actualización de un torneo, validar la transformación de JSON al objeto de dominio Tournament y validar que el valor que se le transfiera a TournamemtDelegate es el esperado. Simular ID no encontrado en el Delegate y validar que la respuesta de esta función sea HTTP 404

Pruebas mínimas esperada en TournamentDelegate

- Al método que procesa la creación de torneo, validar que el valor que se le transfiera a TournamentRepository es el esperado. Simular una inserción válida y que la respuesta de esta función sea el ID generado
- Al método que procesa la creación de torneo, validar que el valor que se le transfiera a TournamentRepository es el esperado. Simular una inserción fallida y que la respuesta de esta función sea un error o mensaje usando https://en.cppreference.com/w/cpp/utility/expected.html
- Al método que procesa la búsqueda de un torneo por ID, validar que el valor que se le transfiera a TournamentRepository es el esperado. Simular el resultado con un objeto y validar valores del objeto
- Al método que procesa la búsqueda de un torneo por ID, validar que el valor que se le transfiera a TournamentRepository es el esperado. Simular el resultado nulo y validar nullptr
- Al método que procesa la búsqueda de torneos. Simular el resultado con una lista de objetos de TournamentRepository
- Al método que procesa la búsqueda de torneos. Simular el resultado con una lista vacía de TournamentRepository
- Al método que procesa la actualización de un torneo, validar la búsqueda de TournamentRepository por ID, validar el valor transferido a Update. Simular resultado exitoso
- Al método que procesa la actualización de un torneo, validar la búsqueda de TournamentRepository por ID. Simular resultado de búsqueda no exitoso y regresar error o mensaje usando https://en.cppreference.com/w/cpp/utility/expected.html

Grupos

Los siguientes endpoints serán programados en la clase de GroupController

URI: /tournaments/<TOURNAMENT-ID>/groups

Método: GET

Cuerpo:

Respuesta:

- HTTP: 200
- HTTP: 404

- En caso de TEAM-ID no exista en la base de datos

```
[
	{
		“name” : “Group A”,
		“teams”: [
			{“id” : “team-id”, “name” : “team name”}
		]
	},
	{
		“name” : “Group B”,
		“teams”: [
			{“id” : “team-id”, “name” : “team name”}
		]
	}
]
```

URI: /tournaments/<TOURNAMENT-ID>/groups

Para crear un grupo, de manera opcional se pueden pasar la lista de grupos, estos deben de validarse de acuerdo al tipo de torneo que se este administrando, e. g., para el Torneo con formato Mundial el número de equipos por grupo es de 4

Método: POST

Cuerpo:

```
{
	“name” : “Group A”,
	“teams”: [
		{“id” : “team-id”, “name” : “team name”}
	]
}
```

Respuesta:

- HTTP: 201
- HTTP: 404
    - En caso de TOURNAMENT-ID no exista en la base de datos

URI: /tournaments/<TOURNAMENT-ID>/groups/GROUP-ID

Para agregar un equipo a un grupo en particular, dicho servicio debe de validar que el número de equipos sea el adecuado para cada grupo. Opcional es que al agregar un equipo cuando un grupo ya este al máximo se agregue automáticamente al siguiente grupo

Método: POST

Cuerpo:

```
{“id” : “team-id”, “name” : “team name”}
```

Respuesta:

- HTTP: 201
- HTTP: 404
    - En caso de TOURNAMENT-ID no exista en la base de datos

Evento al agregar un equipo (procesamiento de fondo para generar partidos)

Al agregar un equipo o equipos, generar un evento por cada equipo generado que contenga la información necesario para identificar el equipo; el consumidor del evento se encargará de generar los partidos de acuerdo a las reglas del torneo en cuestión.

Pruebas mínimas para Group controller

- Al método que procesa la creación de grupo para un torneo, validar la transformación de JSON al objeto de dominio Group y validar que el valor que se le transfiera a GroupDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 201
- Al método que procesa la creación de grupo para un torneo, validar la transformación de JSON al objeto de dominio Group y validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular error de inserción en la base de datos y validar la respuesta HTTP 409
- Al método que procesa la búsqueda de un grupo por ID y torneo por ID, validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular el resultado con un objeto y validar la respuesta HTTP 200

- Al método que procesa la búsqueda de un grupo por ID y torneo por ID, validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular el resultado nulo y validar la respuesta HTTP 404
- Al método que procesa la actualización de un grupo en un torneo, validar la transformación de JSON al objeto de dominio Group y validar que el valor que se le transfiera a GroupDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 204
- Al método que procesa la actualización de un grupo en un torneo, validar la transformación de JSON al objeto de dominio Group y validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular ID no encontrado en el Delegate y validar que la respuesta de esta función sea HTTP 404
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a GroupDelegate es el esperado. Validar que la respuesta de esta función sea HTTP 204
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular que el equipo no exista y el resultado sea HTTP 422
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar la transformación de JSON al objeto de dominio Team y validar que el valor que se le transfiera a GroupDelegate es el esperado. Simular que el grupo este lleno el resultado sea HTTP 422

Pruebas mínimas para Group delegate

- Al método que procesa la creación de grupo para un torneo, simular lectura de Tournament y validar que el valor que se le transfiera a GroupRepository es el esperado, validar que un evento es generado. Validar que la respuesta sea ID del grupo
- Al método que procesa la creación de grupo para un torneo, simular lectura de torneo por medio de TournamentRepository, validar Group se le transfiera a GroupRepository es el esperado. Simular error de inserción cuando ya existe el grupo usando https://en.cppreference.com/w/cpp/utility/expected.html
- Al método que procesa la creación de grupo para un torneo, simular lectura de torneo por medio de TournamentRepository, validar Group se le transfiera a GroupRepository es el esperado. Simular error de inserción cuando ya se llego

al número máximo de equipos usando https://en.cppreference.com/w/cpp/utility/expected.html

- Al método que procesa la búsqueda de un grupo por ID y torneo por ID, validar que el valor que se le transfiera a GroupRepository es el esperado. Simular el resultado con un objeto
- Al método que procesa la búsqueda de un grupo por ID y torneo por ID, validar que el valor que se le transfiera a GroupRepository es el esperado. Simular el resultado nulo
- Al método que procesa la actualización de un grupo en un torneo, validar el valor de Group transferido a GroupRepository es el esperado. Simular la actualización válida

- Al método que procesa la actualización de un grupo en un torneo, validar el valor de Group transferido a GroupRepository es el esperado. Simular ID no encontrado y regresar el valor usando https://en.cppreference.com/w/cpp/utility/expected.html
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar Team se le transfiera a TeamRepository y GroupRepository es el esperado, validar que un mensaje sea publicado. Simular la respuesta de ID
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar Team sea transferido a TeamRepository. Simular que el equipo no exista, regresar el valor https://en.cppreference.com/w/cpp/utility/expected.html
- Al método que procesa la agregar un equipo a un grupo en un torneo, validar Team sea transferido a TeamRepository y GroupRepository. Simular que el grupo este lleno, regresar el valor https://en.cppreference.com/w/cpp/utility/expected.html

Consumidor de evento al agregar un equipo a un torneo

Este consumidor se encargará de generar los partidos de la primera ronda una vez identificado que el número total de equipos haya concluido. Los partidos serán creados de acuerdo a las reglas particulares de cada formato de torneo.

Consumidor de marcador

Este consumidor tendrá reglas más específicas para cada torneo y aplicará las reglas apropiadas si existe alguna acción automatizada que se pueda generar, e. g., en el

torneo de doble eliminación, crear el juego del árbol 2 para los que perdedores del grupo 1.

Torneos

Eliminación sencilla

Todos los torneos a excepción de Doble Eliminación, se genera un formato de eliminación sencilla en donde 8 equipos calificarán de acuerdo con las siguientes condiciones para formato cada torneo, en esta parte del torneo los empates no están permitidos y un ganador debe de ser identificado.

Round Robin

En este torneo de 1 grupo de 32 equipos, se juega una ronda en donde 1 equipo juega un partido contra los demás equipos. Al finalizar la ronda de juegos se tabulan los equipos el 8 juegan la segunda fase que será de eliminación sencilla hasta que quede un campeón.

El orden de la tabulación será por juegos ganados, puntos a favor y puntos en contra.

Limitar el marcador de los partidos entre 0 y 10, con empate como resultado válido.

En la fase de eliminación sencilla no se permiten empates

El torneo consta con un total de 240 partidos en la primera ronda, con 7 partidos de eliminación sencilla.

Una vez tabulados los equipos al finalizar la primera fase, la eliminación sencilla se organizará de la siguiente manera

1	vs	16
4	vs	13
5	vs	12
8	vs	9

7	vs	10
6	vs	11
3	vs	14
2	vs	15

Grupos (Formato Mundial)

Este torneo consta de 8 grupos de 4 equipos cada uno sumando 32 equipos en total, la primera fase consta de juegos round robin intergrupales de 1 vuelta, equipos de un grupo no compiten con equipos de otro grupo en esta primera fase. 2 equipo con mejor tabulación son los que pasan a la ronda de eliminación sencilla.

El orden de la tabulación será por juegos ganados, puntos a favor y puntos en contra.

Limitar el marcador de los partidos entre 0 y 10, con empate como resultado válido.

En la fase de eliminación sencilla no se permiten empates

Supongamos que los grupos son A, B, C, D, E, F, G y H; los juegos de eliminación sencilla serán de la siguiente manera:

A1	vs	H2
B2	vs	G1
C1	vs	F2
D2	vs	E1

D1	vs	E2
C2	vs	F1
B1	vs	G2
A2	vs	H1

Grupos con Wildcard (NFL)

Este torneo consta de 8 grupos de 4 equipos cada uno sumando 32 equipos en total, la primera fase consta de juegos round robin intergrupales de 1 vuelta, y 1 juego contra un equipo de los otro grupos, e. g., el equipo A1 (equipo 1 del grupo A) tendrá juegos vs A2, A3 y A4 (juegos integrupales), además de estos también tendrá los juegos vs B1, C1, D1, E1, F1, G1, H1; A2 jugará sus juegos contra A1, A3, A4 , B2, C2, D2, E2, F2, G2, H2 y asi con todos los equipos. No está demás mencionar que por ser una sola ronda el partido A2 vs B2 es igual que B2 vs C2. Los partidos extra grupales como A1 vs B2 no deben de ser creados.

El orden de la tabulación será por juegos ganados, puntos a favor y puntos en contra.

Limitar el marcador de los partidos entre 0 y 10, con empate como resultado válido.

En la fase de eliminación sencilla no se permiten empates

Hacer una separación lógica de 2 grupos identificando una conferencia. En la clase de grupo agregar una propiedad que identifique la conferencia a la que se pertenece. Los grupos A, B, C, D identificarlos con un valor arbitrario, los grupos E, F, G, H con otro valor diferente.

7 equipos de cada conferencia califican a la ronda de eliminación sencilla.

El equipo con record más alto de cada grupo pasan a la siguiente ronda. Obteniendo los primeros 8 equipos (4 por conferencia)

El equipo con el record más alto de cada conferencia pasa “BYE”, es decir no tiene juego en la primera ronda de eliminación sencilla

Los siguientes 6 equipos se obtienen ordenando por record y puntos, el resto de los equipos (excluyendo los ganadores de cada grupo) y se seleccionan los primeros 3 de cada conferencia.

1 2 vs 7 3 vs 6 4 vs 5	1 2 vs 7 3 vs 6 4 vs 5

Doble eliminación

El formato de este tipo de torneo es similar a uno de eliminación sencilla con la condición adicional de que un equipo tiene que perder 2 veces para quedar eliminado.

Este torneo consiste en 2 árboles de eliminación, el primer árbol inicialmente tendrá los 32 equipos, los partidos se organizarán mientras se agreguen los equipos al grupo A, de esta forma la primera ronda de partidos se tiene la expectativa de Equipo 1 vs Equipo 2, Equipo 3 vs Equipo 4 … etc.

El segundo árbol se va formando a partir de los resultados del primer grupo, el primer equipo que se agrega es el equipo que NO gana del partido entre Equipo 1 vs Equipo 2; su contrincante será el equipo que NO gane del partido Equipo 3 vs Equipo 4.

Los equipos que van perdiendo en el primer árbol van siendo agregados como los siguientes oponentes del segundo árbol. Los equipos que pierdan en el segundo árbol son eliminados.

La final consta del equipo que ganó todos sus partidos del primer árbol; el segundo equipo es resultado de quien “sobreviva” en el segundo árbol. Para determinar el ganador del torneo, potencialmente se juegan 2 partidos; el equipo que viene del primer

árbol solo tiene que ganar 1 partido, el equipo que viene del segundo árbol tiene que ganar ambos partidos.